CLAIMS

What is claimed:

1. A method for distributing cryptographic requests to a plurality of cryptographic devices comprising:

receiving a cryptographic request;

determining the lowest T(N); and

sending the request to a cryptographic device with the lowest T(N).

2. The method of claim 1 further comprising:

determining whether there is a second cryptographic request; and

responsive to a determination that there is a second cryptographic request, determining a new lowest T(N); and

sending the second cryptographic request to a device with the new lowest T(N).

3. The method of claim 1 further comprising:

setting N equal to 1;

setting T(N) equal to 0;

setting Q(N) equal to 0;

determining whether there is another device to query; and

responsive to a determination that there is another device to query, setting N equal to N plus 1 and returning to the step of setting T(N) equal to 0.

4. The method of claim 1 further comprising:

setting CT equal to CST; and

updating all estimated QI completion times.

5. The method of claim 1 further comprising:

setting ET from an ET table;

determining whether a QI is the only QI in queue; and

responsive to a determination that the QI is the only QI, setting the QI timestamp to CT.

6. The method of claim 1 further comprising:

setting N equal to 1;

determining whether Q(N) is empty;

responsive to a determination that Q(N) is empty, determining whether there is another device;

responsive to a determination that there is another device, setting N equal to N plus 1 and returning to the step of determining whether Q(N) is empty;

computing t where t is the time a request in a QI at the top of a queue has been processing, by subtracting the time stamp from CT;

subtracting t from the QI's estimated completion time;

determining whether the new estimated completion time is less than or equal to zero;

responsive to a determination that the new estimated completion time is less than or equal to zero, setting the estimated time to Z percent of the original estimated time.

7. The method of claim 1 further comprising:

identifying a cryptographic device associated with a QI with a completed request;

determining whether there are more QI's in queue for the cryptographic device; and

responsive to a determination that there are more QI's in queue for the cryptographic device, calculating the current system time and assigning the current system time to the next QI's timestamp

8. A programmable apparatus for balancing the load of requests for cryptographic operations sent to a plurality of identical cryptographic devices comprising:

a computer having a processor, a memory, a plurality of PCI buses, and a plurality of cryptographic devices connected to said PCI buses;

a cryptographic API installed on said computer;

a loading balancing program in said cryptographic API;

a estimated completion time subroutine in said load balancing program;

wherein, said estimated completion time subroutine directs said processor to determine a lowest T(N); and

wherein, responsive to determining a lowest T(N), sending a request for a cryptographic operation to a device with the lowest T(N).

9. The programmable apparatus of claim 8 further comprising an initialization subroutine in said load balancing program that directs said processor to set N equal to 1, set T(N) equal to 0, set Q(N) equal to 0.

10. The programmable apparatus of claim 8 further comprising a subroutine in the load balancing program that sets CT equal to CST and updates all estimated QI completion times.

11. The programmable apparatus of claim 8 further comprising a subroutine in the load balancing program that sets ET from an ET table, determines whether a QI is the only QI in queue, and responsive to a determination that the QI is the only QI, sets the QI timestamp to CT.

12. The programmable apparatus of claim 8 further comprising a subroutine in the load balancing program that computes t where t is a time a request in a QI at a top of a queue has been processing, by subtracting a time stamp from CT.

13. The programmable apparatus of claim 8 further comprising a subroutine in the load balancing program that sets N equal to 1, determines whether Q(N) is empty, responsive to a determination that Q(N) is empty, determines whether there is another device, responsive to a determination that there is another device, sets N equal to N plus 1, computes t, where t is a time that a request in a QI at a top of a queue has been processing, by subtracting a time stamp from CT, subtracts t from the QI's estimated completion time, determines whether the new estimated completion time is less than or equal to zero, and responsive to a determination that the new estimated completion time is less than or equal to zero, sets the estimated time to Z percent of the original estimated time.

14. The programmable apparatus of claim 8 further comprising a subroutine in the load balancing program identifies a cryptographic device associated with a QI with a completed request, determines

whether there are more QI's in queue for the cryptographic device, and responsive to a determination that there are more QI's in queue for the cryptographic device, calculating the current system time and assigning the current system time to the next QI's timestamp

15. A computer readable memory for causing a computer to balance the load of requests for cryptographic operations sent to a plurality of cryptographic devices comprising:

a memory;

a load balancing program stored in said memory;

the memory, so configured by said load balancing program, responsive to receiving a request for a cryptographic operation, causes the computer to determine a lowest T(N), and send the cryptographic request to a cryptographic device with the lowest T(N).

16. The computer readable memory of claim 15 wherein the load balancing program comprises an initialization subroutine in said load balancing program that causes said computer to set N equal to 1, set T(N) equal to 0, set Q(N) equal to 0.

17. The computer readable memory of claim 15 wherein the load balancing program comprises a subroutine in the load balancing program that sets CT equal to CST and updates all estimated QI completion times.

18. The computer readable memory of claim 15 wherein the load balancing program comprises a subroutine in the load balancing program that sets ET from an ET table, determines whether a QI is

the only QI in queue, and responsive to a determination that the QI is the only QI, sets the QI timestamp to CT.

19. The computer readable memory of claim 15 wherein the load balancing program comprises a subroutine in the load balancing program that computes t where t is a time a request in a QI at a top of a queue has been processing, by subtracting a time stamp from CT.

20. The computer readable memory of claim 15 wherein the load balancing program comprises a subroutine in the load balancing program that sets N equal to 1, determines whether Q(N) is empty, responsive to a determination that Q(N) is empty, determines whether there is another device, responsive to a determination that there is another device, sets N equal to N plus 1, computes t, where t is a time that a request in a QI at a top of a queue has been processing, by subtracting a time stamp from CT, subtracts t from the QI's estimated completion time, determines whether the new estimated completion time is less than or equal to zero, and responsive to a determination that the new estimated completion time is less than or equal to zero, sets the estimated time to Z percent of the original estimated time.

21. The computer readable memory of claim 15 wherein the load balancing program comprises a subroutine in the load balancing program identifies a cryptographic device associated with a QI with a completed request, determines whether there are more QI's in queue for the cryptographic device, and responsive to a determination that there are more QI's in queue for the cryptographic device, calculating the current system time and assigning the current system time to the next QI's timestamp

22. A computer implemented process to balance the load of requests for cryptographic operations sent to a plurality of cryptographic devices, comprising: using a computer, performing the following series of steps:

receiving a cryptographic request;

setting N equal to 1;

setting T(N) equal to 0;

setting Q(N) equal to 0;

determining whether Q(N) is empty;

responsive to a determination that Q(N) is empty, determining whether there is another device;

responsive to a determination that there is another device, setting N equal to N plus 1 and returning to the step of determining whether Q(N) is empty;

computing t where t is the time a request at the top of the queue has been processing by subtracting the time stamp from CT;

subtracting t from the request's estimated completion time;

determining whether the new estimated completion time is less than or equal to zero;

responsive to a determination that the new estimated completion time is less than or equal to zero, setting the estimated time to Z percent of the original estimated time;

responsive to a determination that the new estimated time is greater than zero, determining whether there is another device to query;

responsive to determining that there is another device to query, returning to the step of

determining whether Q(N) is empty; and

identifying the device associated with the completed request;

determining whether there are more QIs in queue;

responsive to a determination that there are more QIs in queue, calculating the current system time and assigning the current system time to the next QI's timestamp;

setting CT equal to CST;

updating all estimated completion times;

determining the device with the lowest T(N); and

sending the cryptographic request to a device with the lowest T(N)

23. The computer implemented process of claim 22 further comprising:

determining whether there is another device to query;

responsive to a determination that there is another device to query, setting N equal to N plus 1 and returning to the step of setting T(N) equal to 0.

determining whether there is a second cryptographic request;

responsive to a determination that there is a second cryptographic request, determining a new lowest T(N); and

sending the second cryptographic request to a cryptographic device with the lowest T(N).